НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет
Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050101 “ Комп’ютерні науки ”

на тему: «Аналіз виконання показників бюджету на основі онтологічного підходу з використанням платформи Neo4j»

Виконав: студент IV курсу, групи ТР-51

Зінькевич Богдан Романович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник Дацюк Оксана Антонівна _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає зозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний
Кафедра автоматизації проектування енергетичних процесів і систем
Рівень вищої освіти перший рівень
Напрям підготовки 6.050101 “ Комп’ютерні науки”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту
Зінькевича Богдана Романовича
(прізвище, ім’я, по батькові)

1. Тема роботи: «Аналіз виконання показників бюджету на основі онтологічного підходу з використанням платформи Neo4j»
керівник роботи Дацюк Оксана Антонівна, старший викладач
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 201__р. № __

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи мова програмування Java, середовище IntelliJ IDEA, бібліотека Thymeleaf, платформа Neo4j

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) розробити алгоритми аналізу на основі онтологічного підходу, здійснити програмну реалізацію розроблених методів.

5. Перелік ілюстративного матеріалу архітектура системи, графічне представлення інтерфейсу, приклади роботи програмного модулю
6. Дата видачі завдання ” 10 ” жовтня 2019 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	14.10.2018-23.12.2018	
2.	Розробка архітектури та загальної структури системи	2.02.2019-3.03.2019	
3.	Розробка структур окремих підсистем	4.03.2019-14.04.2019	
4.	Підготовка матеріалів	15.04.2019-18.04.2019	
5.	Програмна реалізація системи	18.04.2019-14.05.2019	
6.	***Захист програмного продукту***	15.05.2019	
7.	Оформлення пояснювальної записки	16.05.2019-3.06.2019	
8.	***Передзахист***	28.05.2019	
9.	Захист	17.06.2019-22.06.2019	

АНОТАЦІЯ

Обсяг дипломної роботи складає 64 сторінки, 15 рисунків і 12 посилань

Метою дипломної роботи є розробка програмного додатку, який дозволить користувачу проводити аналіз виконання показників бюджету на основі онтологій

У ході роботи проаналізовані засоби створення онтологій – Protégé, GraphDB , Neo4j та зроблено порівняльну характеристику. В роботі обґрунтовано використання онтологічного підходу для зберігання даних показників виконання бюджету. В якості подальшого розвитку системи планується створення графічного інтерфейсу для заповнення даних та встановленню між ними зв’язків

Результатом роботи стало створення програмного продукту для аналізу виконання показників бюджету та змогою робити експорт, імпорт даних у форматі Excel.

Ключові слова: онтологія, аналіз бюджету, Protégé, OWL, контекст онтології, RDF, елементи онтології, Neo4j.

ANNOTATION

The purpose of the thesis is to develop a software application that will allow the user to conveniently analyze the budget indicators based ontologies

During the work, the means of creating ontologies - Protégé, GraphDB, Neo4j - have been analyzed and a comparative characteristic has been made. Also, fundamental work was carried out on studying existing methods of budget analysis and proposed alternative in the form of an ontological approach. As a further development of the system it is planned to create a graphical interface for filling data and establishing links between them

The result of the work was the creation of a software product for the analysis of the budget and the ability to export, import data in Excel.

Key words: ontology, budget analysis, Protégé, OWL, ontology context, RDF, elements of ontology, Neo4j.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КПК - Коди програмної класифікації видатків
КБК - Коди бюджетної класифікації
КФК - Коди функціональної класифікації видатків
J2EE - Java Platform, Enterprise Edition
SOAP - Simple Object Access Protocol
CRM - Customer relationship management

ВСТУП

У сучасному світі є необмежений доступ до великої кількості інформації. Швидкий розвиток інформаційних технологій приводить до появи інформаційних систем великого масштабу. Багато аналітичних систем використовують велику кількість різнорідної інформації.

Вже давно існує проблема організації збору потрібної інформації для її подальшого аналізу. До таких задач можна віднести задачу моніторингу виконання показників бюджетів. Фактично, це процес збору та аналізу даних про доходи та видатки, згідно закону України «Про державний бюджет» .

Звіти про виконання державного бюджету на регіональному рівні надходять до Держказначейства України. Автоматизоване введення даних про виконання бюджетів всіх рівнів передбачає використання відповідних форм звітів Державної казначейської служби України, представлених у форматі MS Excel. Звітність про виконання бюджетів всіх рівнів є оперативною, місячною, квартальною та річною. Всі ці звіти проходять етапи збору інформації від окремих підприємств до районної звітності в області, відомства, міністерства. Існують також звітності про виконання окремих державних програм та ін. Розподіл показників є досить розгалудженим, як за територіальними так і за фінансовими показниками, та враховує обсяги фондів та їх цільове призначення зі своїми особливостями та правилами.

Для зберігання та обробки великих об’ємів структурованої інформації традиційно використовують реляційні сховища даних. Але залишається проблема об'єднання даних, отриманих з різних джерел з метою їх подальшої обробки

Вирішенням проблеми формалізації даних може бути онтологічний підхід. Такий підхід досить часто використовується для об’єднання знань предметних областей в єдину інформаційну структуру .

Онтологія - це один із методів візуалізації та побудови зручних графів для дослідження даних. Ця технологія забезпечує хорошу взаємодію різних специфікацій метаданих за допомогою спільної семантики, структури та синтаксису. Мова веб-онтологій або OWL додає більш потужні інструменти моделювання до

RDF і RDFS. OWL забезпечує перевірку на узгодженість та на відповідність певним умовам. Наприклад, чи є якісь логічні невідповідності або чи існують класи, які не можуть мати екземплярів. Також OWL забезпечує класифікацію об’єктних типів.

Нажаль, реляційні бази даних, які традиційно використовують для задач такого типу, не мають графічної візуалізації зв'язків між класами предметної області. Аналіз виконання показників бюджету зручно проводити маючи графічну візуалізацію зв'язків та залежностей між окремими даними. Використання онтології якраз допоможе вирішити цю проблему. Онтологія надає такі переваги для реалізації поставленої задачі:

- візуалізація графу даних;
- аналіз за допомогою предикатів;
- валідація вхідних параметрів;
- представлення ієрархічної структури на різних рівнях вкладення;
- зручне підтримання актуалізації даних.

Самі ж звітні дані знаходяться в таблицях, які пов’язані з класами онтології. За допомогою фреймворка «Ontop» можна під’єднати онтологію до таблиць бази даних. В результаті можливе виконання запитів до онтології, які автоматично будуть отримувати дані з таблиць. Для того, щоб фреймворк «Ontop» правильно генерував і виконував запити до бази даних потрібно як можна точніше описати всі властивості об’єктів і типів даних з онтології і створити всі потрібні «маппінги» для всіх властивостей.

Таким чином можна проводити автоматичний збір та обробку інформації на основі звітів про хід виконання держбюджету, що сприяє контролю за ефективністю використання коштів.

1. ЗАДАЧА АНАЛІЗУ ПОКАЗНИКІВ БЮДЖЕТУ НА ОСНОВІ ОНТОЛОГІЧНОГО ПІДХОДУ

Метою розробки є створення програмного додатку, який дозволяє оператору зручно проводити аналіз виконання показників бюджету для будь-яких адміністративних одиниць. Збільшити зручність користування шляхом створення програмного інтерфейсу

Призначенням даного програмного засобу є надання зручного інтерфейсу для імпорту, експорту даних формату Excel та аналізу бюджету

Програмний засіб розробляється для комп'ютерів під керівництвом системи Windows XP/7/10.

Вхідні дані: файл формату Excel із даними бюджету по адміністративних одиницях.

Вихідні дані: проведений аналіз бюджету, експорт файлу у формат Excel .

Розглянути можливість створення власного програмного продукту для автоматизації процесу збору інформації та аналізу виконання бюджету.

Необхідними можливостями, які має забезпечувати модуль, повинні бути:

- завантаження файлу Excel з комп'ютера користувача;
- створення онтології із даними користувача;
- аналіз виконання показників бюджету;
- можливість перегляду даних онтології у структурованому форматі;
- можливість відкриття тієї ж самої онтології за допомогою Neo4j;
- можливість побудови діаграм у Excel;
- підрахунок відсотку виконання бюджету;
- можливість аналізувати як групу адміністративних одиниць, так і кожну

2. ДОСЛІДЖЕННЯ ПРОБЛЕМИ АНАЛІЗУ ВИКОНАННЯ ПОКАЗНИКІВ БЮДЖЕТУ

2.1 Опис предметної області

Моніторинг всіх рівнів фінансових показників виконання бюджету - це процес послідовного збору та аналізу даних про доходи, видатки та показники виконання бюджетів всіх рівнів , фінансування за типом боргового зобов'язання та кредитора та включення виконання захищених статей видатків ДБУ, кредитування за КБК, КФК, КПК. Моніторингом бюджету в межах країни займається спеціальний державний орган – Рахункова палата України. Рахункова палата – це орган фінансово-бюджетного контролю, який утворюється Верховною Радою України, підпорядкований і підзвітний їй. Працює самостійно і незалежно від інших органів держави.

Основні обов'язки рахункової палати:

- організація і здійснення контролю за своєчасним виконанням видаткової частини Державного бюджету України, витрачанням бюджетних коштів, у тому числі коштів загальнодержавних цільових фондів, за обсягами, структурою та їх цільовим призначенням здійснення контролю за утворенням і погашенням внутрішнього і зовнішнього боргу України, визначення ефективності та доцільності видатків державних коштів, валютних та кредитно-фінансових ресурсів;
- контроль за фінансуванням загальнодержавних програм економічного, науково-технічного, соціального і національно-культурного розвитку, охорони довкілля;
- контроль за дотриманням законності щодо надання Україною позик і економічної допомоги іноземним державам, міжнародним організаціям, передбачених у Державному бюджеті України;

- контроль за законністю та своєчасністю руху коштів Державного бюджету України та коштів позабюджетних фондів в установах Національного банку України та уповноважених банках;
- аналіз встановлених відхилень від показників Державного бюджету України та підготовка пропозицій про їх усунення, а також про удосконалення бюджетного процесу в цілому;
- регулярне інформування Верховної Ради України, її комітетів про хід виконання Державного бюджету України та стан погашення внутрішнього і зовнішнього боргу України, про результати здійснення інших контрольних функцій;
- виконання інших завдань, передбачених для Рахункової палати чинним законодавством України.

До складу Рахункової палати входять Голова Рахункової палати та члени Рахункової палати: Перший заступник і заступник Голови, головні контролери та Секретар Рахункової палати.

Працівники рахункової палати щомісячно обробляють і аналізують дані понад 7.8 тис. нових показників, тому необхідне програмне забезпечення, яке б суттєво підвищувало оперативність та ефективність роботи при здійсненні функцій контролю і аналізу стану виконання державного бюджету. і аналізу стану виконання державного бюджету. Перевагою ПЗ автоматизованого введення казначейських звітів є суттєве скорочення часу. Завдяки цим засобам введення даних за доходами та видатками бюджетів всіх рівнів з обсягом понад 100000 записів/на місяць (рядків) у базі знань на основі онтологічного підходу виконується за 1 - 2 хвилини. Ручне введення такої кількості показників з наступною їх верифікацією потребує декілька робочих днів.

Для здійснення своєї діяльності Рахункова палата має апарат. Структуру і штатний розпис апарату Рахункової палати затверджує Колегія Рахункової палати за поданням Голови Рахункової палати в межах бюджетних коштів, передбачених на її утримання.

2.2 Існуючі рішення

Існуючі програми для аналізу бюджету вузькопрофільні і не доступні для широкого доступу. Однією з таких систем є «Система моніторингу фінансових показників виконання бюджетів всіх рівнів», розроблена у 2013 році.

Для збору і аналізу показників бюджету використовуються Excel звіти у форматі. :

Звіт про дебіторську заборгованість за КЕК (тис.грн.)

КЕК	КВК		КПК	Назва	На початок року	Всього на звітну дату	Прострочена на звітну дату	Списана на звітну дату
0000	ВСЬОГО				8 368 981,00	6 563 775,90	1 172 832,66	8 357,15
0002	ВИДАТКИ				8 368 981,00	6 563 775,90	1 172 832,66	8 357,15
	11	Апарат Верховної Ради України			16,17	181,36	0,00	0,00
	30	Державне управління справами			1 027,18	12 742,78	61,60	0,00
	41	Господарсько-фінансовий департамент Секретаріату Кабінету Міністрів			100,58	94,30	0,00	0,00
	50	Державна судова адміністрація України			6 040,97	5 683,90	4 215,14	0,00
	60	Верховний Суд України			46,30	21,07	0,00	0,00
	75	Вищий адміністративний суд України			37,42	25,06	0,00	0,00
	80	Конституційний Суд України			23,88	0,77	0,00	0,00
	90	Генеральна прокуратура України			1 702,35	1 356,45	0,00	0,00
	100	Міністерство внутрішніх справ України			1 111 858,00	692 388,09	108 426,05	2,98
	110	Міністерство енергетики та вугільної промисловості України			208,09	213,13	1,49	0,00
	120	Міністерство економічного розвитку і торгівлі України			1 061,15	185,17	19,77	0,00
	121	Міністерство економічного розвитку і торгівлі України (загальнодержавні			8,40	0,00	0,00	0,00
	140	Міністерство закордонних справ України			53 736,43	51 928,87	0,00	0,00
	170	Державний комітет телебачення і радіомовлення України			293,09	137,81	108,68	0,00
	180	Міністерство культури України			83 122,49	68 606,92	1 654,39	0,00
	210	Міністерство оборони України			2 898 143,67	2 699 168,33	476 621,50	8 321,02
	220	Міністерство освіти і науки України			1 240,09	3 345,21	225,46	0,00
	230	Міністерство охорони здоров'я України			2 999 296,40	1 702 032,80	129 755,52	0,00
	240	Міністерство екології та природних ресурсів України			10 714,19	7 449,60	602,93	0,00
	250	Міністерство соціальної політики України			46 690,22	101 008,48	34 976,90	1,22

ПЗ «Моніторинг показників виконання бюджетів всіх рівнів» дає можливість досліджувати бюджетні процеси за такими напрямами:

- моніторинг виконання державного бюджету;
- моніторинг виконання зведеного бюджету;
- моніторинг виконання місцевих бюджетів;
- моніторинг виконання місцевих бюджетів у розрізі регіонів;
- моніторинг заборгованості бюджетних установ.

Аналіз фінансових показників здійснюється за такими напрямками: доходи та видатки бюджетів (державного бюджету, місцевих бюджетів, зведеного бюджету), включаючи доходи за кодами бюджетної класифікації, видатки бюджетів всіх рівнів за КФК, КПК, КЕКВ, показники фінансування і заборгованості, виконання захищених статей видатків ДБУ, кредитування за КБК, КФК, КПК, а також фінансування за типом боргового зобов'язання та кредитора. Аналіз показників здійснюється у розрізі року, кварталу, місяцю року, а також у розрізі регіонів.

Дані ФЕП зберігаються на сервері БД РП. Система має близько 40 незв'язаних між собою та ненормалізованих таблиць. 10 з цих таблиць призначені для тимчасового зберігання даних, підготованих для OLAP аналізу.

Нажаль, розширення методів аналізу даних в даній ситемі обмежене ненормалізованою структурою БД та відсутністю зв'язків між окремими класами

даних. Доповнення таких зв’язків до реляційної БД може суттєво збільшити об’єм даних та знизити швидкодію системи.

Такий підхід вимагає великої кількості людських ресурсів, зменшує точність і надійність ведення звітів, так як виникає ризик людської помилки і неточного введенення даних або використання не правильної формули для аналізу. А також із кількістю накопиченою інформації ускладнюється зберігання та структуризація звітів. Перевагами такого підходу є те, що користувачів не потрібно навчати користуватися програмою Excel і дешевизна підходу, тому що не потрібна підтримка програмних інженерів для спеціалізованої системи

2.3 Висновки до розділу

У даному розділі описано визначення та функції Розрахункової палати України. Розглянуті існуючі способи. організації зберігання та обробки даних щодо виконання показників бюджету. Наведені їх переваги та недоліки. Постійне редагування та запис даних забирає багато часу у користувача . Цей недолік суттєво сповільнює роботу по аналізу даних робить існуючий підхід неконкурентоспроможним.

3. ЗАСОБИ РОЗРОБКИ

Основним середовищем розробки було середовище Intellij IDEA

Для розробки безпосередньо алгоритмів використовувалась мова програмування Java 8

Для графічного інтерфейсу був використана мова розмітки HTML5

Для передачі даних було між сервером і веб-інтерфейсів використано шаблон темплейтів Thymeleaf

Для відображення деревоподібної структури елементів онтології використовувалася платформа Neo4J

3.1 Мова програмування Java 8

Java є основою практично для всіх типів додатків і загальним стандартом для розробки і поширення мобільних додатків, ігор, веб-контенту та корпоративного програмного забезпечення. Технологія Java протестована, вдосконалена, розширена і перевірена учасниками спільноти розробників Java, архітекторами і ентузіастами. Java дозволяє розробляти високопродуктивні портативні програми практично на всіх комп'ютерних платформах. Доступність додатків в різнорідних середовищах дозволяє компаніям надавати більш широкий спектр послуг, сприяє підвищенню продуктивності, рівня взаємодії і спільної роботи кінцевих користувачів і істотного зниження вартості спільного володіння корпоративними і споживчими додатками.

Основні властивості платформи Java:

- написання програмного забезпечення на одній платформі і його запуск практично на будь-який інший платформі
- створення програм, що працюють в веб-браузері і мають доступ до веб-служб
- розробка додатків на стороні сервера для форумів в Інтернеті, магазинів, опитувань, обробки форм HTML і багато іншого
- об'єднання додатків або служб з використанням мови Java для створення високоспеціалізованих додатків або служб
- створення багатофункціональних і ефективних додатків для мобільних телефонів, віддалених процесорів, мікроконтролерів, бездротових модулів, датчиків, шлюзів, споживчих продуктів і практично будь-яких інших категорій електронних пристроїв

Мова Java була обрана для реалізація даного програмного продукту тому що вона об’єктно-орієнтована, має хорошу документація, кросплатформенна і виправдала свою швидкодію при високих навантаження.

3.2 Середовище розробки IntelliJ IDEA

IntelliJ IDEA – це інтегроване середовище розробки для різних мов програмування. Середовище розроблене компанією JetBrains. Дизайн середовища орієнтовано на продуктивність праці програмістів, дозволяючи їм сконцентруватися на розробці функціональності, тоді як IntelliJ IDEA бере на себе виконання рутинних операцій, таких як генерація простого коду для побудови моделі класів. Також IntelliJ IDEA дозволяє легко інтегруватися із системами контролю версій, та локально налаштовувати деплой програмного продукту на сервер

3.3 Графічний інтерфейс HTML5

HTML5 є найновішим стандартом HTML. Цей термін представляє дві різні концепції:

- це нова версія мови HTML, з новими елементами, атрибутами і поведінкою
- це великий стек технологій, котрі надають більшого різноманіття та потужності Веб-сайтам та додаткам

Переваги HTML5 в порівнянні з HTML:

- Семантика: дозволяє описати якомога точніше з чого складається ваш контент.
- Зв'язок: дозволяє вам взаємодіяти з сервером новим та інноваційним шляхом.
- Оффлайн та сховище: надає можливість веб-сторінкам зберігати дані на стороні клієнта локально та оперувати оффлайном ефективніше.
- 2D/3D графіка та ефекти: дозволяє безліч презентаційних варіантів.
- Ефективність та інтеграція: надає більшу оптимізацію швидкості та кращого використання заліза (hardware) комп'ютера.

- Доступ до пристрою: дозволяє використання різноманітних пристроїв вводу/виводу
- Стилі: Дозволяє авторам створювати більш складні і витончені теми.

3.4 Бібліотека Thymeleaf

Thymeleaf - сучасний серверний механізм Java-шаблонів для веб-і автономних середовищ, здатний обробляти HTML, XML, JavaScript, CSS і навіть простий текст. Основною метою Thymeleaf є створення елегантного і зручного способу шаблонізації. Щоб досягти цього, Thymeleaf грунтується на концепції Natural Templates, щоб впровадити свою логіку в файли шаблонів таким чином, щоб цей шаблон не впливав на відображення прототипу дизайну. Це покращує комунікацію в команді і зменшує розрив між дизайнерсько-програмістськими групами. Thymeleaf також був розроблений з самого початку з урахуванням стандартів Web, особливо HTML5, що дозволяє вам створювати повністю відповідають стандарту шаблону

3.5 Графічна платформа Neo4j

Графічна платформа приймає новий підхід для збереження даних. Вона розширює можливості реляційних баз даних, так як граф із з’язками відображає реальні сутність об’єкту.

Графічна платформа Neo4j побудована навколо бази даних Neo4j:

- База даних Neo4j підтримує програмні транзакції ACID та аналітику графів
- Графічна аналітика допомагає відображати дані бази даних у зручному для користувача форматі
- Інтеграція даних прискорює перетворення реляційних даних у графіки
- Мова запитів графіка Cypher є інструментом для аналізу великих масивів даних
-
- Архітектура предметної області лежить в основі і підтримує відношення між зв’язками графа

3.5 Висновки до розділу

У даному розділі розглянуті інструменти та технології, які були необхідні для розробки програмного інструментального засобу. Розроблений додаток написано на мові об’єктно-орієнтованого програмування Java з використанням шаблону Thymeleaf платформи Java 8. Розробка проводилася з використанням середовища IntelliJ IDEA та принципів розробки дизайну графічного інтерфейсу. Обробки файлу онтології передбачає використання платформи Neo4j. Для зменшення етапів створення нових елементів онтології доцільно застосовувати графічні елементи HTML5

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Архітектура та шаблони проектування

Розроблений додаток надає можливість, імпортувати і експортувати дані у форматі файла Excel, робити аналіз виконання бюджету по адміністративних одиницях за певний період часу, переглядати і доповнювати онтологію у середовищі програмного продукту neo4j.

4.1.1 Архітектура додатку

Програмний додаток розроблявся з використанням мови програмування Java платформи Spring за шаблоном Thymeleaf. Для проведення аналізу використовується платформа Neo4j(Рисунок 4.1).

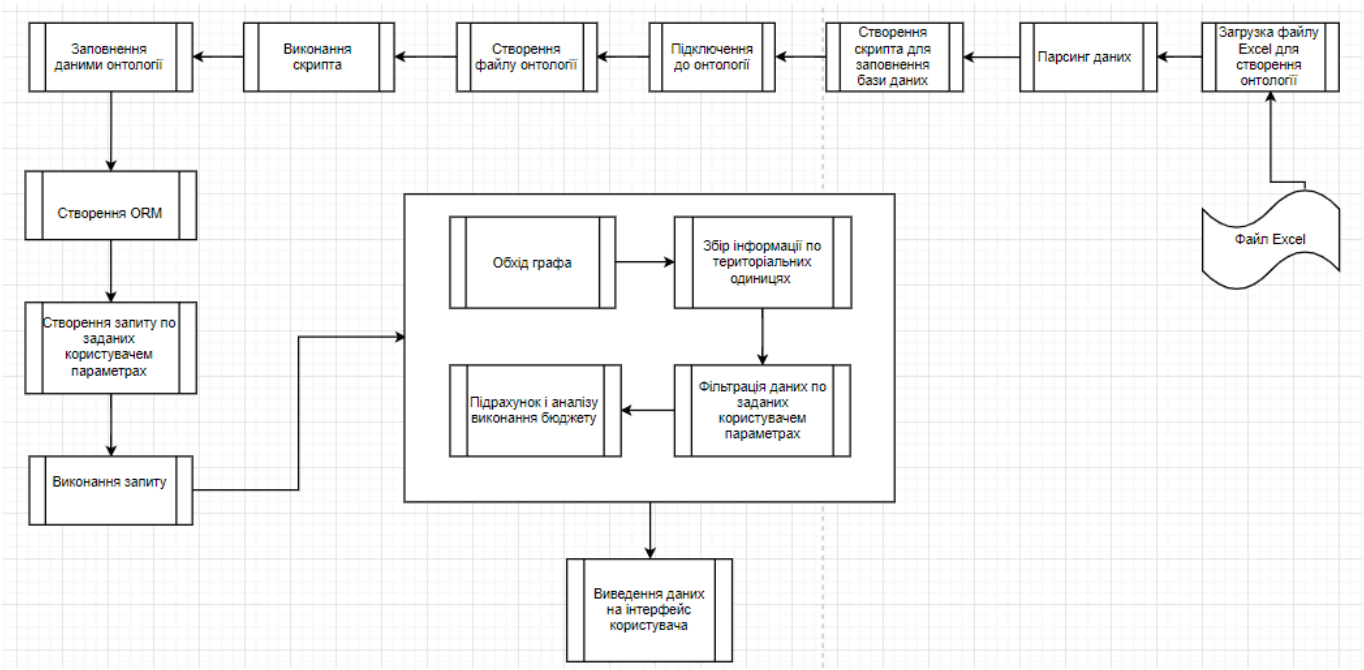


Рисунок 4.1 – Структура програмного засобу

Розроблений програмний засіб складається з наступних основних функціональних частин:

— завантаження файлу Excel з комп’ютера користувача;

— створення онтології із даними користувача;

— аналіз виконання показників бюджету;

— можливість перегляду даних онтології у структурованому форматі;

— можливість відкриття тієї ж самої онтології за допомогою Neo4j;

— можливість побудови діаграм у Excel;

— підрахунок відсотку виконання бюджету;

— можливість аналізувати як групу адміністративних одиниць, так і кожну окремо. Кожна частина вирішує відповідне завдання або проблему, при розробці власного програмного додатку аналізування бюджету.

В результаті розробки, проект додатку має наступний вигляд (Рисунок 4.2):

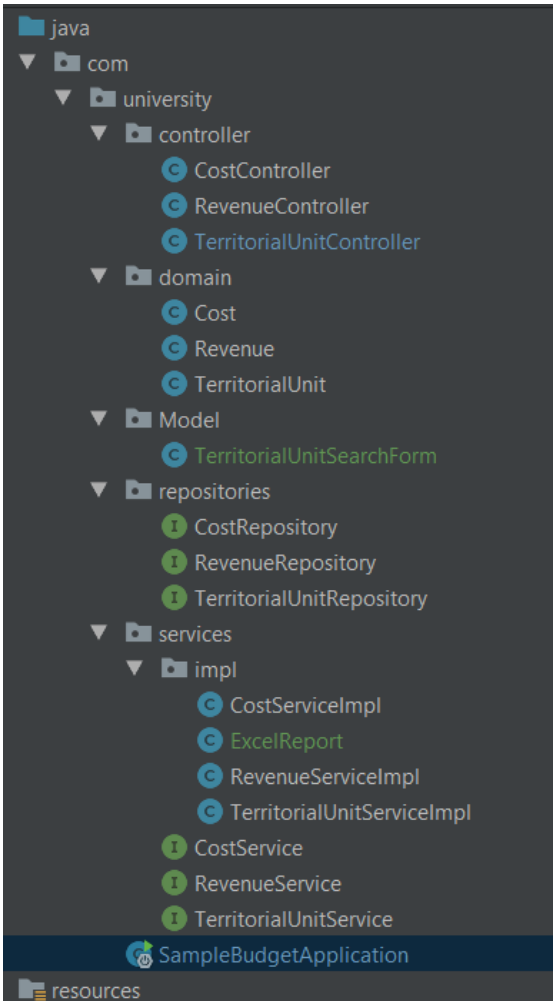


Рисунок 4.2 – Дерево проекту програмного додатку

В процесі користування програмою виконується побудова графу, який точно відображає процес аналізу бюджету. Заповнення даного графа можливо як і скриптом, написаним досвідченим користувачем, так і імпортом даних із формату Excel. Тому якщо доводить обробляти великі масиві інформації, то імпорт даних із формату Excel збереже багато людських ресурсів і допоможе оптимізувати роботу спеціалістів.

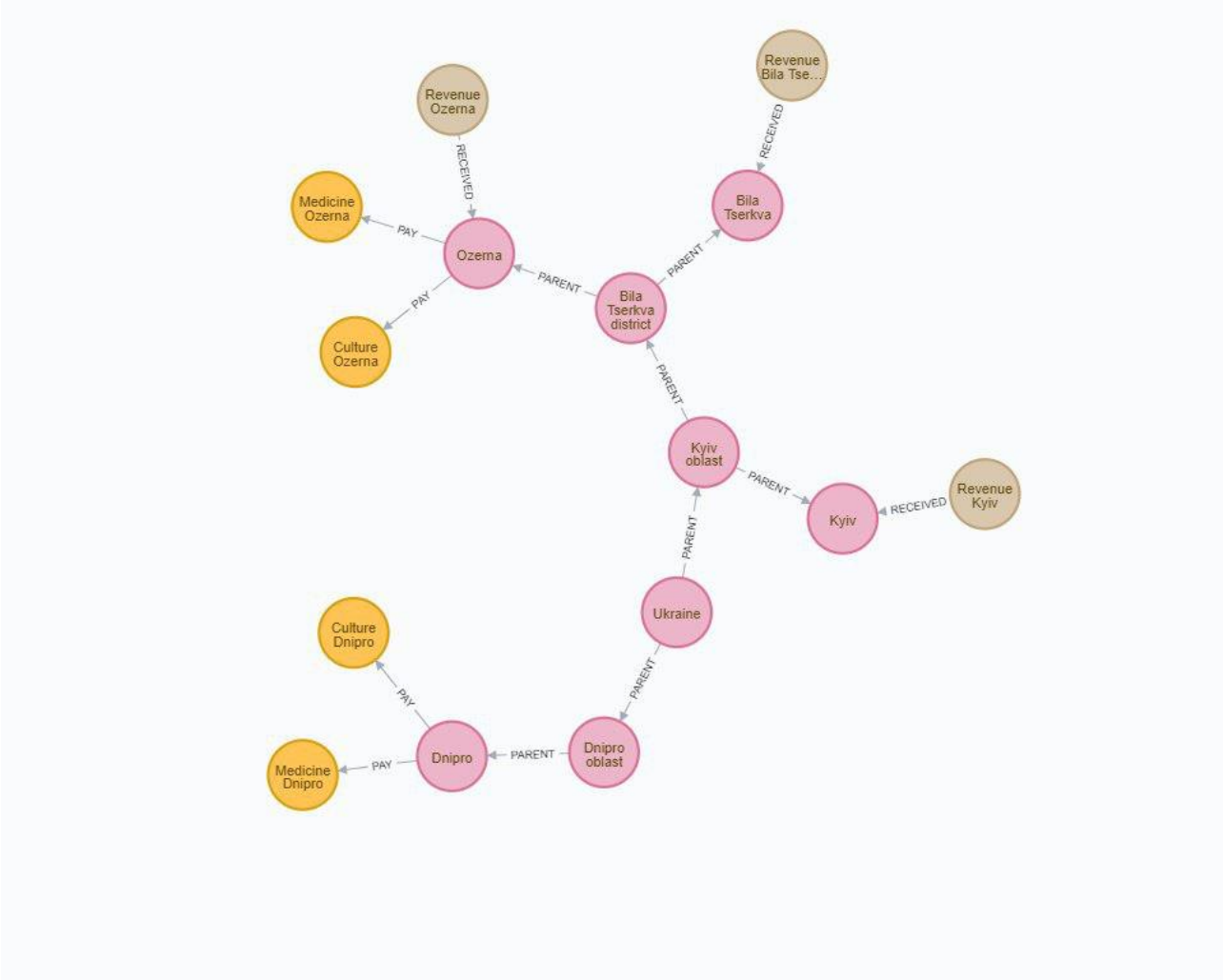


Рисунок 4.3 – Відображення графа онтології

Діагра́ма кла́сів — статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення

деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм. Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

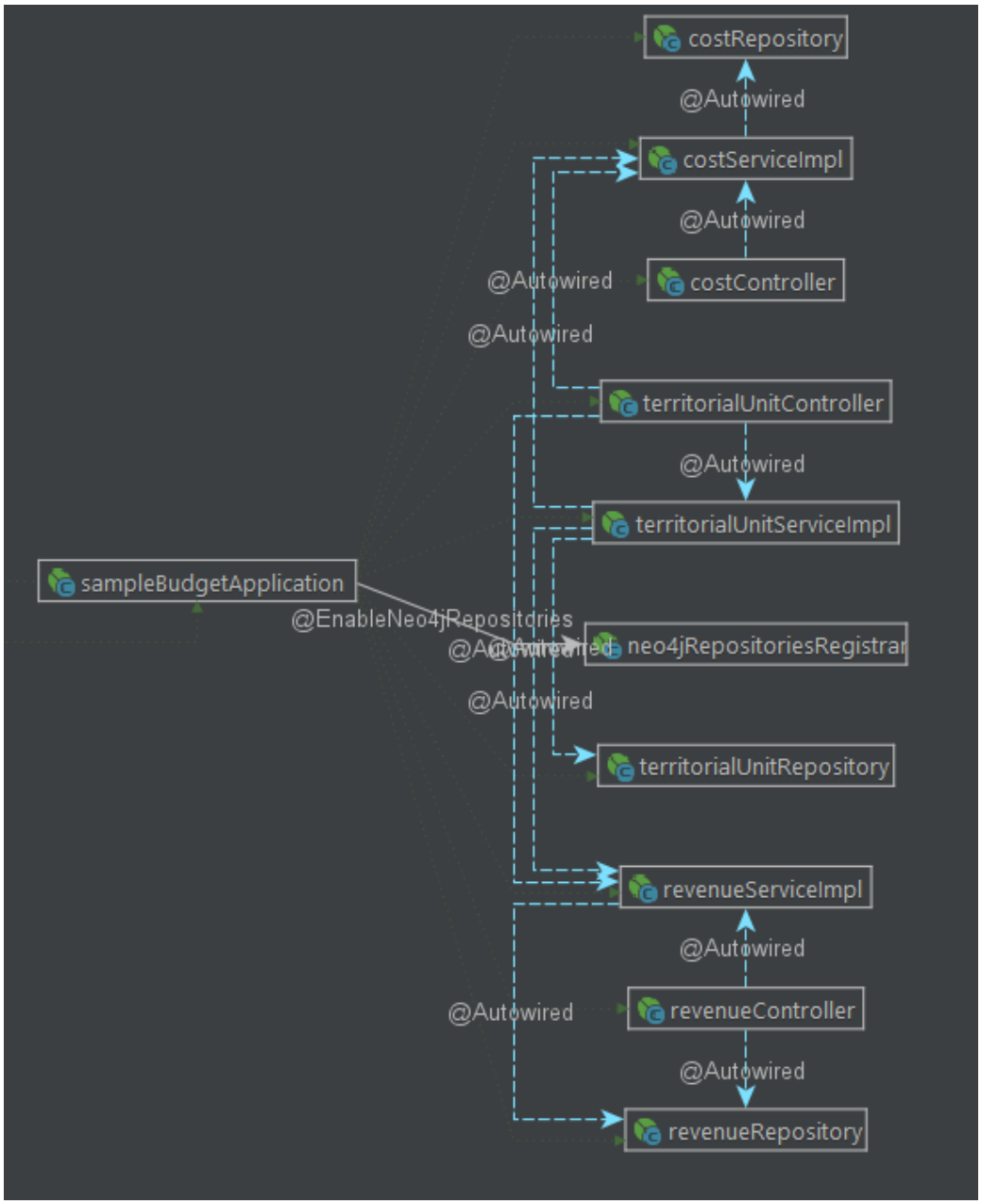


Рисунок 4.4 – Відображення діаграми класів

Асоціація показує, що об'єкти однієї сутності (класу) пов'язані з об'єктами іншої сутності. Якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі

випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарною. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів; вони називаються n-арнимі.

Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами. Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій. Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це «обличчя», яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Ви можете явно позначити роль, яку клас грає в асоціації. Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді. Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: «нуль або одиниця» (0..1), «багато» (0 .. *), «одиниця або більше» (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає «будь-яке число об'єктів, крім 2 і 5».

Агрегація — проста асоціація між двома класами, яка відображає структурне відношення між рівноправними сутностями, коли обидва класи знаходяться на одному концептуальному рівні, і жоден з них не важливіший за решту. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має вищий ранг (ціле) і складається з декількох менших за рангом (частин). Ставлення

такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і її зображують як просту асоціацію з незафарбованим ромбом з боку «цілого». Графічно агрегація представлена порожнім ромбом на блоці класу, і лінією, яка проведена від цього ромба до класу, що міститься в ньому.

Композиція — більш суворий варіант агрегації. Відома також як агрегація за значенням. Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбованим ромбиком.

Імпорт із програми у формат Excel потрібен для того, щоб змогти зручно обмінювати даними та будувати діаграми

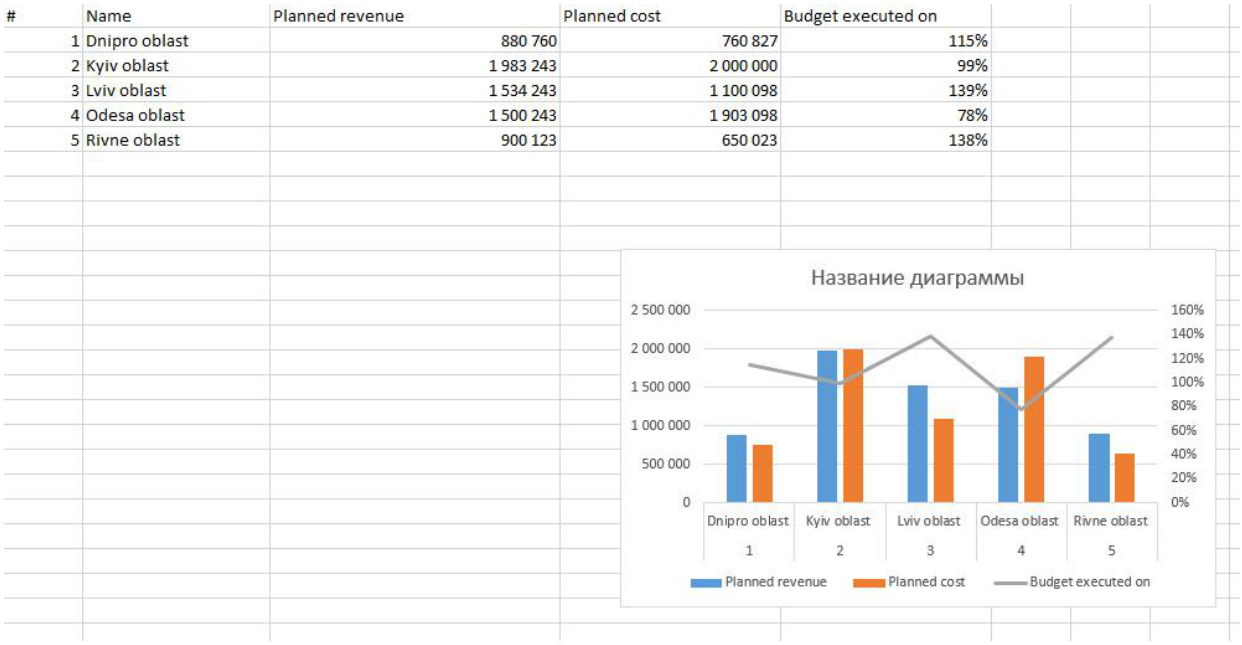


Рисунок 4.4 – Відображення звіту у Excel

4.1.2 Шаблон Factory

Фабричний метод — це породжувальний патерн проектування, який визначає загальний інтерфейс для створення об’єктів у суперкласі, дозволяючи підкласам змінювати тип створюваних об’єктів. Патерн Фабричний метод пропонує відмовитись від безпосереднього створення об’єктів за допомогою оператора new, замінивши його викликом особливого фабричного методу. Об’єкти все одно будуть створюватися за допомогою new, але робити це буде фабричний метод.

Переваги

- Позбавляє клас від прив’язки до конкретних класів продуктів.
- Виділяє код виробництва продуктів в одне місце, спрощуючи підтримку коду.
- Спрощує додавання нових продуктів до програми.
- Реалізує принцип відкритості/закритості.

Недоліки

- Може призвести до створення великих паралельних ієрархій класів, адже для кожного класу продукту потрібно створити власний підклас творця.

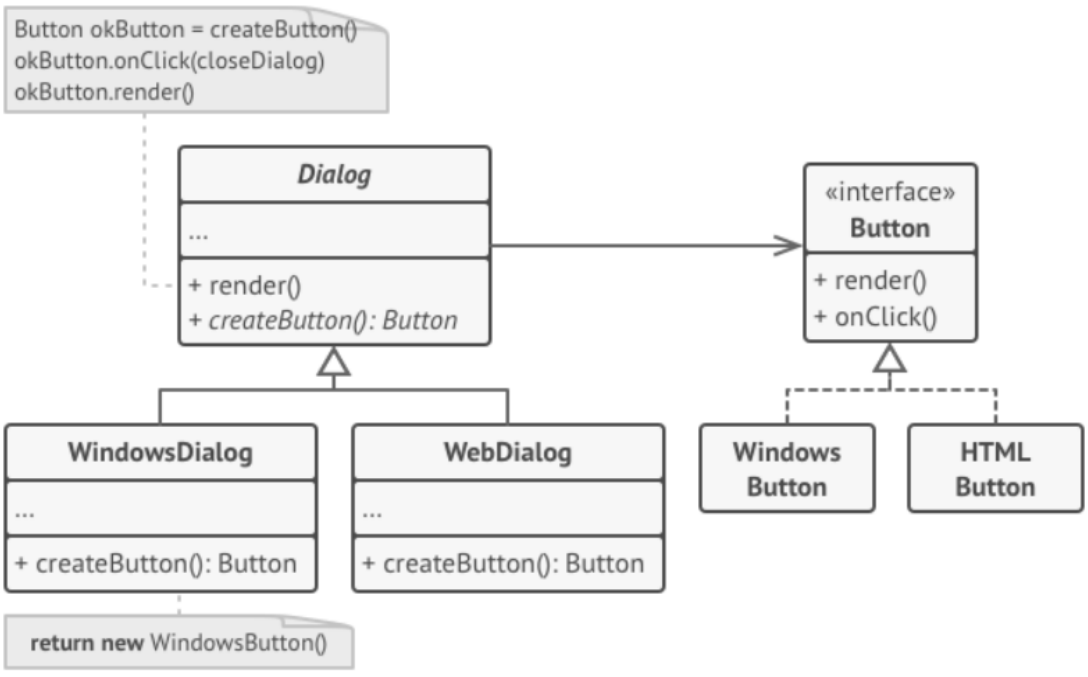


Рисунок 4.5 – Схема зв’язків між класами проекту

4.1.3 Шаблон Singleton

Шаблон Singleton відноситься до групи твірних та є одним з найвідоміших шаблонів проектування програмного забезпечення. Його використовують тоді, коли для деякого класу важливо, щоб існував тільки один екземпляр об’єкту. Також цьому екземпляру необхідно надати глобальну точку доступу.

Вдалим рішенням вважається, коли сам клас контролює свою унікальність. Запити на створення нових об’єктів перехоплюються: якщо екземпляр класу вже був створений, необхідно надати доступ до існуючого об’єкту замість того, щоб створювати додатковий новий екземпляр .

Переваги:

- Гарантує наявність єдиного екземпляра класу.
- Надає глобальну точку доступу до нього.
- Реалізує відкладену ініціалізацію об’єкта-одинака

Недоліки:

- Порушує *принцип єдиного обов’язку класу*.
- Маскує поганий дизайн.
- Проблеми багатопоточності.
- Вимагає постійного створення Mock-об’єктів при юніт-тестуванні.

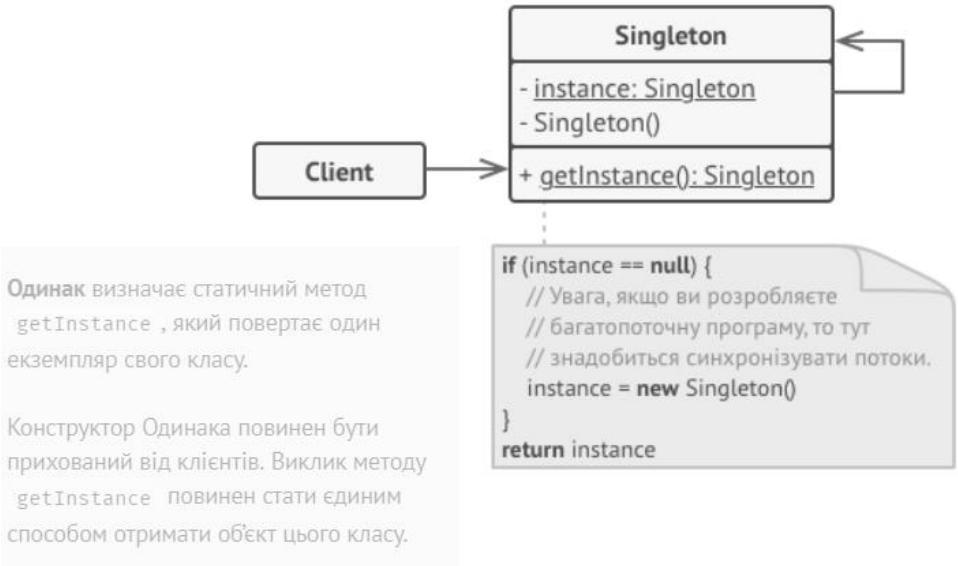


Рисунок 4.6 – Діаграма використання паттерну Singelton

Реалізований шаблон проектування наступним чином. Цей клас не має публічного конструктора, тому єдиним способом отримання його об’єкта є виклик методу getInstance. Цей метод збереже перший створений об’єкт і повертатиме його в усіх наступних викликах.

4.1.4 Шаблон Bridge

Міст — це структурний патерн проектування, який розділяє один або кілька класів на дві окремі ієрархії — абстракцію та реалізацію, дозволяючи змінювати код в одній гілці класів, незалежно від іншої.

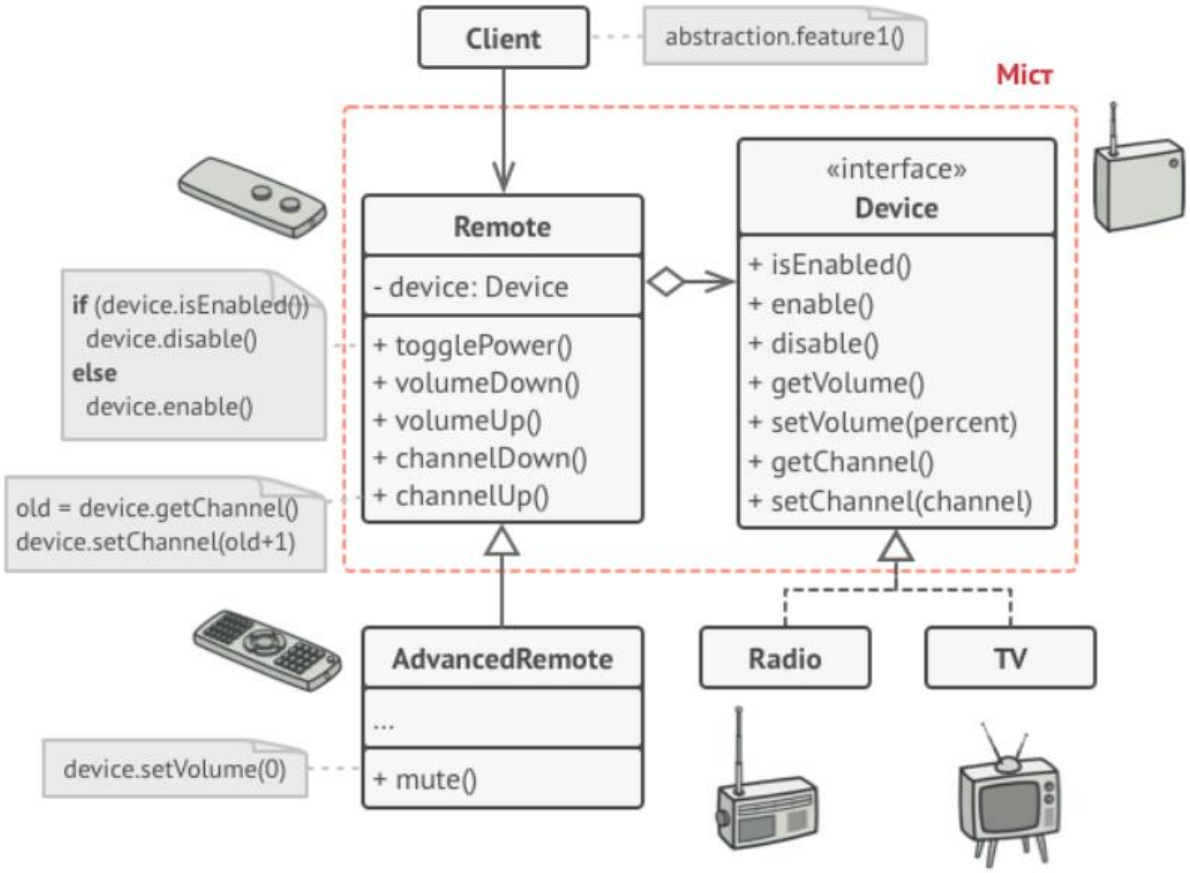


Рисунок 4.7 – Діаграма використання паттерну Bridge

Переваги

- Дозволяє будувати платформо-незалежні програми.
- Приховує зайві або небезпечні деталі реалізації від клієнтського коду.
- Реалізує *принцип відкритості/закритості*.

Недоліки

– Ускладнює код програми внаслідок введення додаткових класів.

4.2 Створення алгоритму аналізу даних

Аналіз даних відбувається на основі графу, який побудований за допомогою онтологічного підходу, тобто відображає різні типи зв’язків. Ця технологія добре підходить для реалізації поставленої задачі, тому що програмний продукт має близьку побудову до реального відображення сутностей.

Граф - це абстрактне уявлення безлічі об'єктів і зв'язків між ними. Графом називають пару (V, E) де V це безліч вершин, а E безліч пар, кожна з яких представляє собою зв'язок (ці пари називають ребрами). Граф може бути орієнтованим або неорієнтованим. В орієнтованому графі, зв'язки є спрямованими (тобто пари в E є впорядкованими, наприклад пари (a, b) і (b, a) це два різні типи зв'язку). У свою чергу в неорієнтованому графі, зв'язок ненаправлений, і тому якщо існує зв'язок (a, b) то значить, що існує зв'язок (b, a).

Фактично дерево – це різновид динамічного списку. Існує багато типів дерев, але я розглядав лише бінарні дерева. Це визвано тим, що такий тип дерев є значно простішим і найбільш зручним у використанні, порівняно з іншими типами дерев

Кожен елемент дерева називається вузлом або листом дерева. Перший вузол дерева (з якого дерево власне починається) називається коренем. Фрагмент дерева разом з вузлом, від якого він починається, називається піддеревом або віткою. Множина всіх вузлів, рівновіддалених від кореня, називається рівнем. Вузол, з якого не починається жодна вітка, називається кінцевим або термінальним вузлом.

Оскільки дерево бінарне, кожен вузол може породжувати два вузли наступного рівня. Породжені вузли є дочірніми по відношенню до вузла, що їх породив. Породжуючий вузол є батьківським по відношенню до своїх дочірніх вузлів.

Батьківський вузол разом із своїми дочірніми складає ланку. Сумарна кількість рівнів дерева називається висотою дерева.

Основними операціями при роботі з деревами є:

-додавання елемента до дерева;

-пошук елемента дерева, що відповідає заданому критерію пошуку;

-сортування елементів дерева;

-вилучення елемента дерева.

Бінарне дерево відображує ієрархію об’єктів територіальних одиниць для аналізу бюджету.

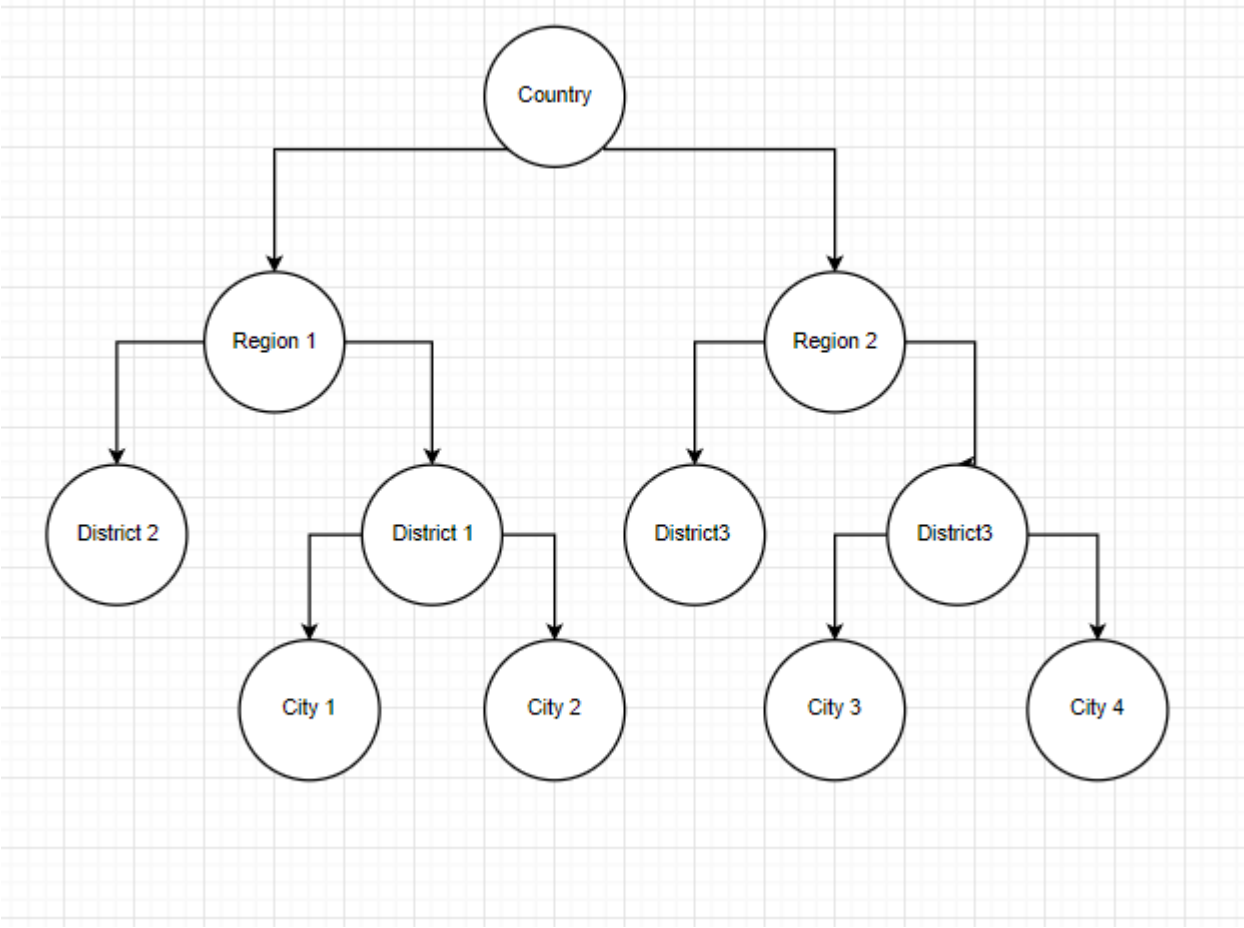


Рисунок 4.7 – Схематична діаграма використання бінарного дерева

4.2.1 Аналіз існуючих інструментів обробки онтологій для аналізу бюджету

Стандартних засобів представлення RDF об’єктів у Java немає, тому актуальним вважаю проаналізувати існуючі засоби розв’язання цієї проблеми.

Розглянемо та порівняємо за наступними критеріями дві бібліотеки: Jena API та Owlim (Таблиця 4.1).

Критерій порівняння	Бібліотека Jena API	Бібліотека Owlim
1. Наявність документації.	Наявність повної документації для користування бібліотекою	Наявна документація повністю описує можливі варіанти застосування бібліотеки. Підібрані приклади використання онтологій в мові Java
2. Простота використання.	Назви методів та властивостей схожі до тих, які описані у специфікації онтологій	Перед використанням необхідно ознайомитись з поняттями та принципами роботи з RDF.
3. Достовірність представлення онтології в пам’яті та запису її у файл.	Структура файлу створеного бібліотекою відповідає структурі онтології, але інколи цю бібліотеку неможливо використовувати, так як вона спотворює структуру файлу	Структура файлу відповідає структурі онтології і бібліотека створює формат онтології який є достовірним
4. Підтримка синтаксисів запису онтології.	RDF/XML Syntax	RDF/XML Syntax OWL/XML Syntax JSON Syntax

Таблиця 4.1 – Порівняння існуючих бібліотек обробки даних онтології

Під час застосування на практиці обох бібліотек було виявлено ряд спільних недоліків:

— Обидві бібліотеки мають проблеми із зчитуванням/запису кирилиці. Через використання в якості властивостей об’єктів класу Uri – інформація кодується 2-ма байтами, а символи Unicode переводяться в escape-послідовності.

— Онтологію легко доповнювати, або видаляти з неї елементи, проте немає механізму зміни характеристик існуючого елементу. Тобто зміна реалізована шляхом створення нового об’єкту та видалення старого.

Окрім того бібліотека Jena API має наступні мінуси:

— неякісний опис бібліотеки – важко зрозуміти яким чином користуватись наявним функціоналом бібліотеки;

— розробник вказує на відсутність реалізації частини функціональності;

— неможливо записати у файл онтології масив пов’язаних об’єктів;

— важко знайти приклади застосування цієї бібліотеки.

Через, наведені вище, недоліки бібліотеку Jena API доцільно застосовувати лише у випадку використання примітивних операцій з онтологією, які не потребують обробки масивів пов’язаних об’єктів. Також використання можливе у випадку аналізу схеми класів бібліотеки Jena API, для проектування власного рішення представлення контексту онтології.

Після практичного застосування та порівняння бібліотек Jena API та Owlim рекомендую використовувати бібліотеку Owlim. Через наявність повної документації, підтримку розробників свого продукту, значну кількість прикладів коду та достовірність представлення онтології вважаю доцільним використовувати саме цю бібліотеку, як більш універсальний засіб створення контексту онтології для мови Java.

4.2.2 Розробка алгоритму обходу графа в глибину

Для реалізації поставленої задачі потрібно обходити граф в глибину з різною глибиною вкладень, які програма буде визначати самостійно, залежно від територіальної одиниці введеної користувачем.

Пошуком в глибину (DFS - depth first search) називається один з методів обходу графа G = (V, E), суть якого полягає в тому, щоб йти "вглиб" поки це можливо. У процесі пошуку в глибину вершин графа присвоюються номери, а ребра позначаються. Обхід вершин графа відбувається згідно з принципом: якщо з поточної вершини є ребра, що ведуть у непройдені вершини, то йдемо туди, інакше повертаємося назад. Пошук в глибину починається з вибору початкової вершини v графа G, яка відразу ж позначається як пройдена. Потім

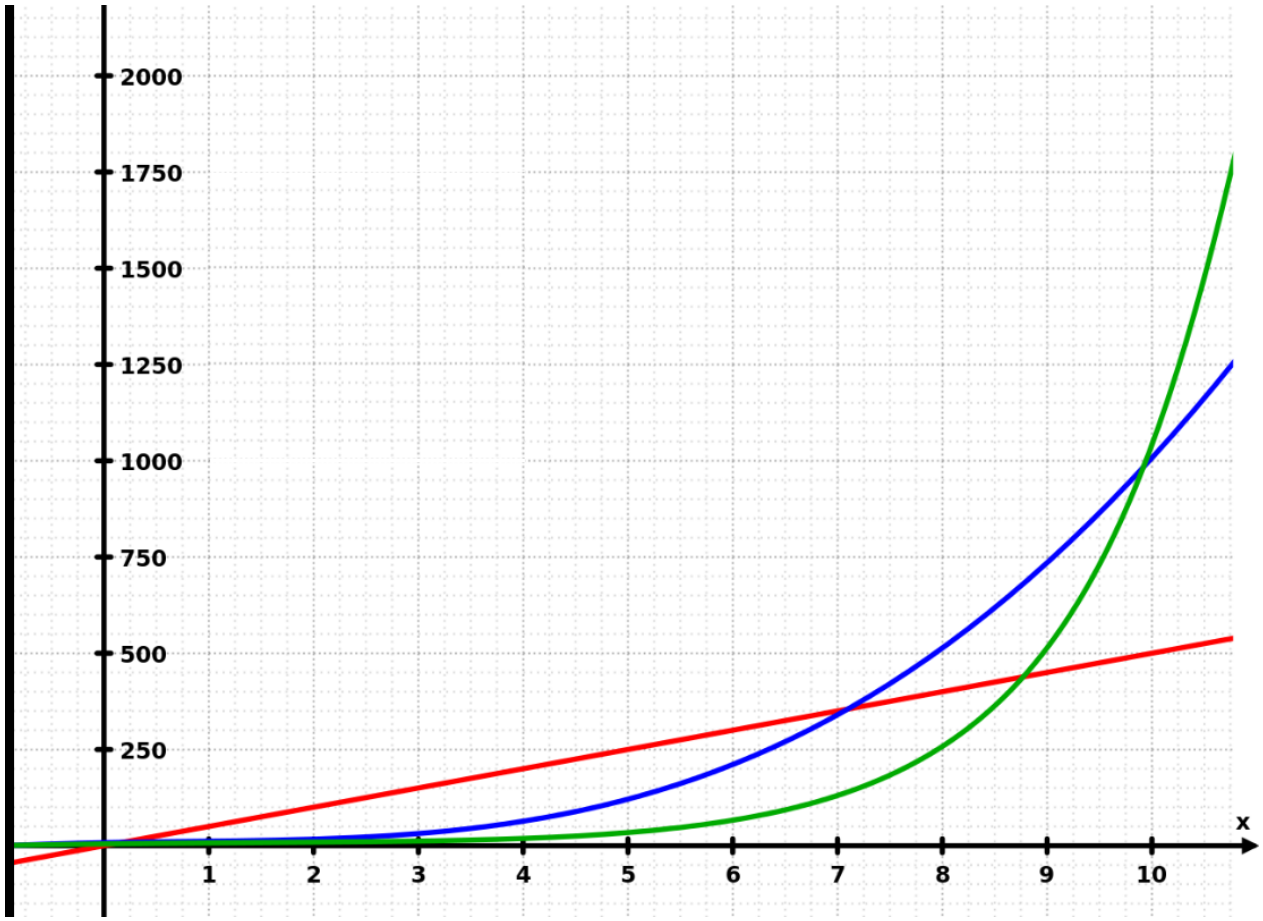
для кожної непоміченими вершини, суміжної з v, рекурсивно викликається пошук в глибину. Коли всі вершини, досяжні з v, будуть позначені, пошук закінчується. Якщо на деякому (не перший) кроці обходу пошук закінчився, але деякі вершини залишаються непоміченими (такий випадок можливий у разі орієнтованого або незв'язною графа), то вибирається довільна з них і пошук повторюється. Процес пошуку триває до тих пір, поки всі вершини графа G не будуть позначені.

Після того, як я отримав потрібну мені інформацію для аналізу бюджету із територіальних одиниць, я проводжу фільтрацію по заданих користувачем параметрах. В даному випадку по початковому місяцю, року та кінцевому. Коли фільтрація завершиться тоді відбувається підрахунок і аналіз отриманих даних. Вся операція пошуку і аналізу займає 0.1 секунду і буде екпоненціально зростати залежно від розміру даних, які потрібно проаналізувати.

Нижче буде наведено порівняльна швидкість алгоритму пошуку на графіках функції:

- лінійна – червона
- синя - квадратична
- зелена – експонціальна

Очевидно, що експонціальна швидкість є задовільною для вирішення задачі, тому що при великих масивах даних, приріст часу обробки не буде критичним



Таблиця 4.2 – Порівняння характеристика графіків

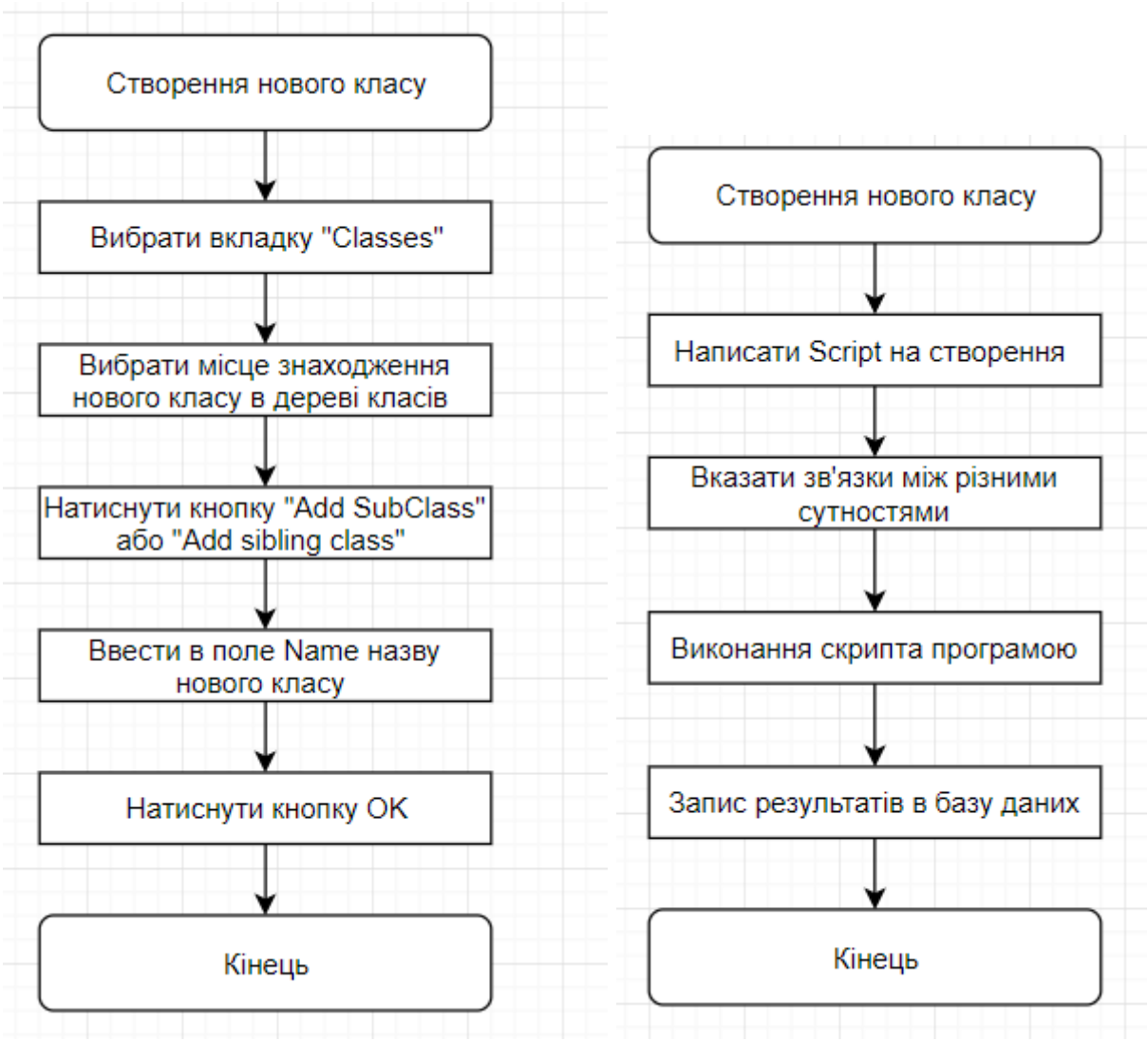
4.3 Зменшення етапів заповнення онтології

Один з недоліків редактора Protégé полягає в тому, що оператор виконує велику кількість операцій під час редагування онтології. Тому метою створення додатку було зменшити для оператора кількість етапів під час введення даних до онтології.

Розглянемо основні етапи, необхідні для того, щоб додати до онтології новий клас в редакторі Protégé. Користувач повинен спочатку розгорнути вкладку “Classes” на головній формі. Якщо редактор запущений вперше після інсталяції, цієї вкладки може не бути. Тоді користувачу необхідно послідовністю дій “Window —> Tabs —> Classes” в меню відобразити вкладку на формі. Далі необхідно розгорнути дерево класів для того, щоб оператор отримав візуалізацію структури наявних в онтології класів. Кількість операцій на цьому етапі залежить від глибини дерева, тобто

оператор натискає на кожний батьківський клас в дереві. Якщо оператор створює декілька класів, для другого та наступних класів, повторювати цей етап не потрібно.

Під час наступного кроку оператор обирає місцезнаходження нового класу в дереві, шляхом натискання на один з його елементів. Для обраного класу, натискаючи на кнопку “Add SubClass” або “Add sibling class” створюється дочірній або сусідній клас відповідно. В редакторі Protégé відкриється модульне вікно для вводу назви нового класу.



а)

Рисунок 4.8 – Етапи створення нового класу: а) в системі Protégé, б) в розробленому додатку

В розробленому програмному засобі оператору не потрібно виконувати наступні етапи:

— розгортати дерево класів;

— натискати спеціальну кнопку для створення дочірнього або сусіднього класу.

Зменшення етапів мною було досягнуто через те, що написання скрипта значно прискорює створення онтологій.

Приклад скрипта на створення онтлогії наведено нижче

```
CREATE
(Ukraine:TerritorialUnit {name: "Ukraine"}),
(DniproOblast: TerritorialUnit {name: "Dnipro oblast" }),
(KyivOblast: TerritorialUnit {name: "Kyiv oblast"}),
(Kyiv:TerritorialUnit {name: "Kyiv"} ),
(Dnipro:TerritorialUnit {name: "Dnipro"} ),
(BilaTserkvaDistrict:TerritorialUnit {name: "Bila Tserkva district"}),
(BilaTserkva:TerritorialUnit {name: "Bila Tserkva"}),
(Ozerna:TerritorialUnit {name: "Ozerna"}),
(Ukraine)-[:PARENT]->(DniproOblast),
(DniproOblast)-[:PARENT]->(Dnipro),
(Ukraine)-[:PARENT]->(KyivOblast),
(KyivOblast)-[:PARENT]->(Kyiv),
(KyivOblast)-[:PARENT]->(BilaTserkvaDistrict),
(BilaTserkvaDistrict)-[:PARENT]->(BilaTserkva),
(BilaTserkvaDistrict)-[:PARENT]->(Ozerna)


CREATE
(RevenueKyiv:Revenue {name: "Revenue Kyiv", value: 3424,  start_date: "2019-01-01"}),
(RevenueBilaTserkva:Revenue {name: "Revenue Bila Tserkva", value: 6565,  start_date: "2019-03-01"}),
(RevenueOzerna:Revenue {name: "Revenue Ozerna", value: 1223, start_date: "2019-03-01"}),
(RevenueKyiv)-[:RECEIVED {value: 19}]->(Kyiv),
(RevenueBilaTserkva)-[:RECEIVED {value: 19}]->(BilaTserkva),
(RevenueOzerna)-[:RECEIVED {value: 19}]->(Ozerna)
```

Рисунок 4.9 – Скрипт на заповнення даних

Подібне рішення не відволікає оператора, тип самим зменшує ризик появи помилок.

4.4 Висновки до розділу

Завдяки використанню пошуку по графу в глибину, підвищена швидкість роботи, що актуально для даної задачі. Також було проведено порівняння швидкості пошуку по графу незалежно від кількості елементів і визначено, що експоненціальний ріст задовольняє вимоги користувачів.

Зменшено кількість етапів занесення інформації до бази даних, завдяки використанню платформи Neo4j, яка реалізує онтологічний підхід і має вбудовану реалізацію інтерпритатора для виконання скриптів . Була проведене порівняння бібліотек роботи з онтологіями в контексті мови програмування - Java, перераховано всі переваги та недоліки і визначено, що Owlim краще задокументована, має приклади

роботи на офіційному сайті та не спотворює файл онтології на відміну від Jena API

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

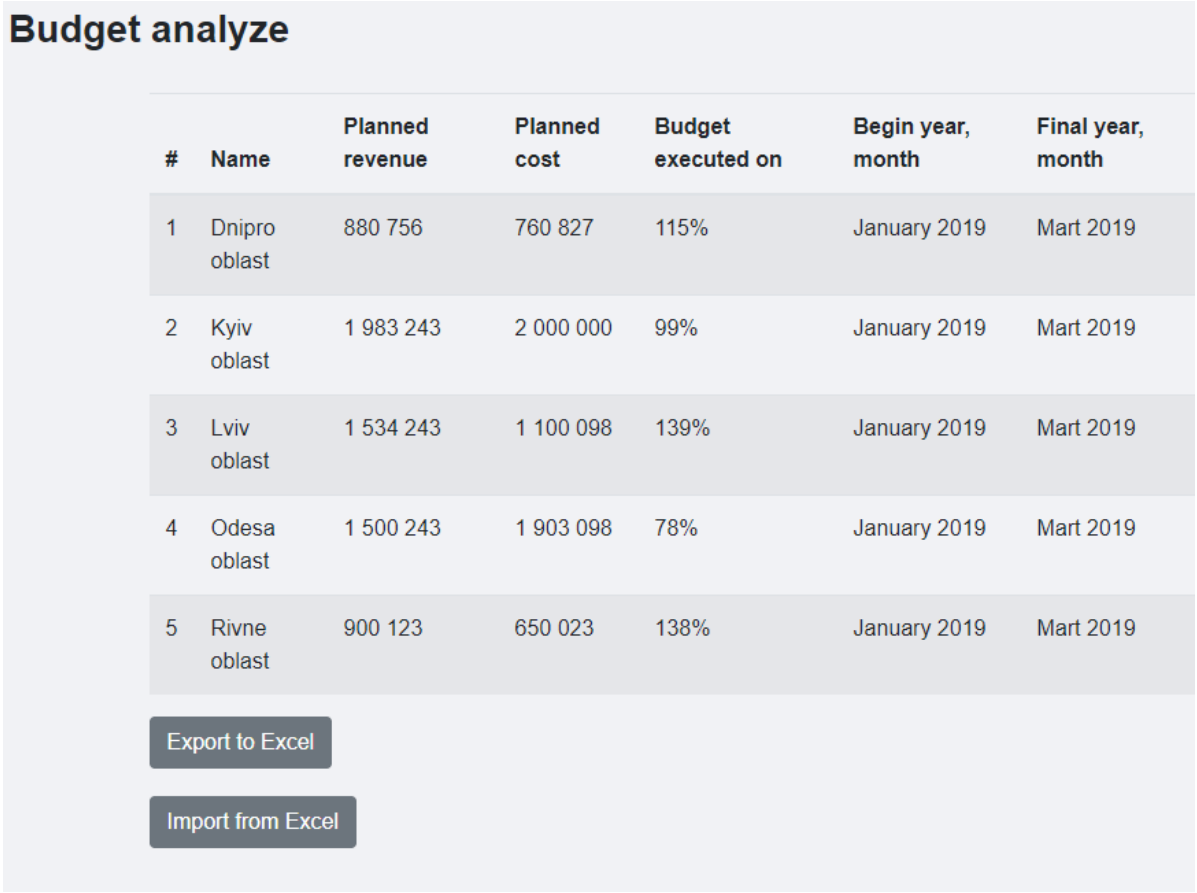
5.1 Опис web-додатку

Проект складається з однієї головної html сторінки, класів із логікою обробки даних та файлів словників назв для локалізації інтерфейсу. Класи згруповані та взаємодіють між собою за шаблоном проектування MVC.

На головній сторінці розташовано кнопки для керування інтерфейсом. Сторінка розроблена з використанням комфортних для очей кольорів. Гама відтінків не є яскравою, що не відволікатиме користувача від роботи і допоможе зосереджено виконувати поставлені задачі.

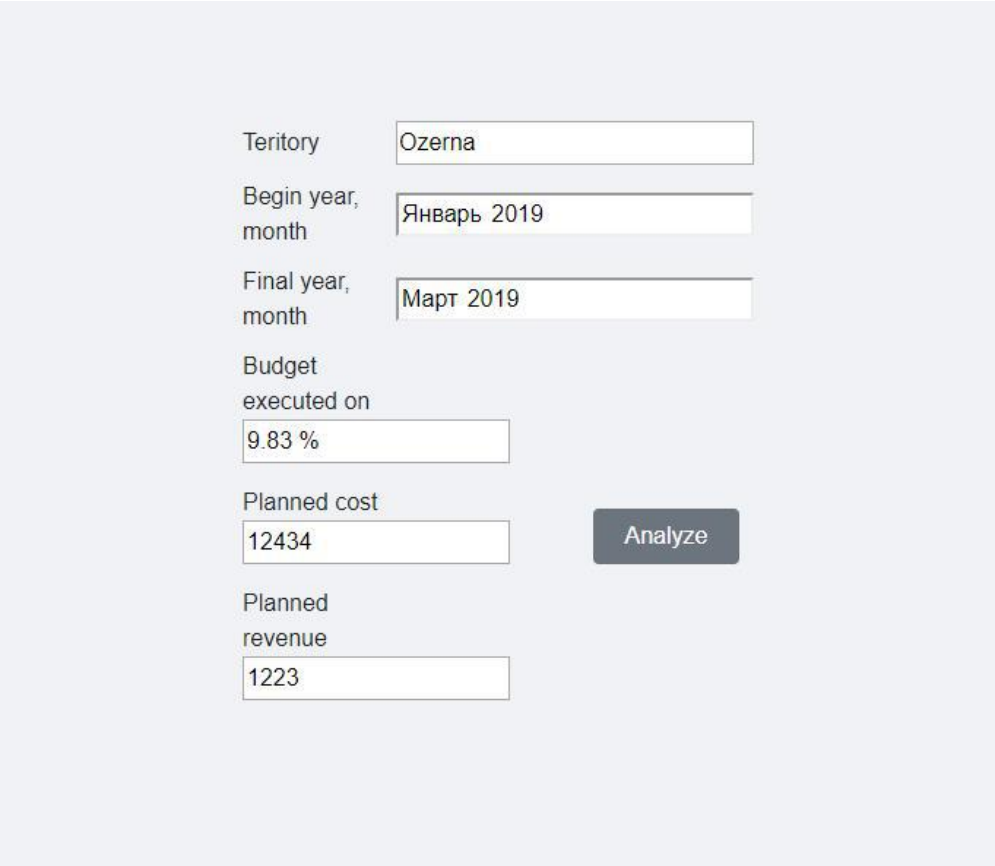
На сторінці існують декілька кнопок:

- analyze – кнопка аналізу бюджету
- import from Excel – кнопка переносу даних в програму з формату Excel
- export to Excel – кнопка переносу даних у формат Excel з бази даних

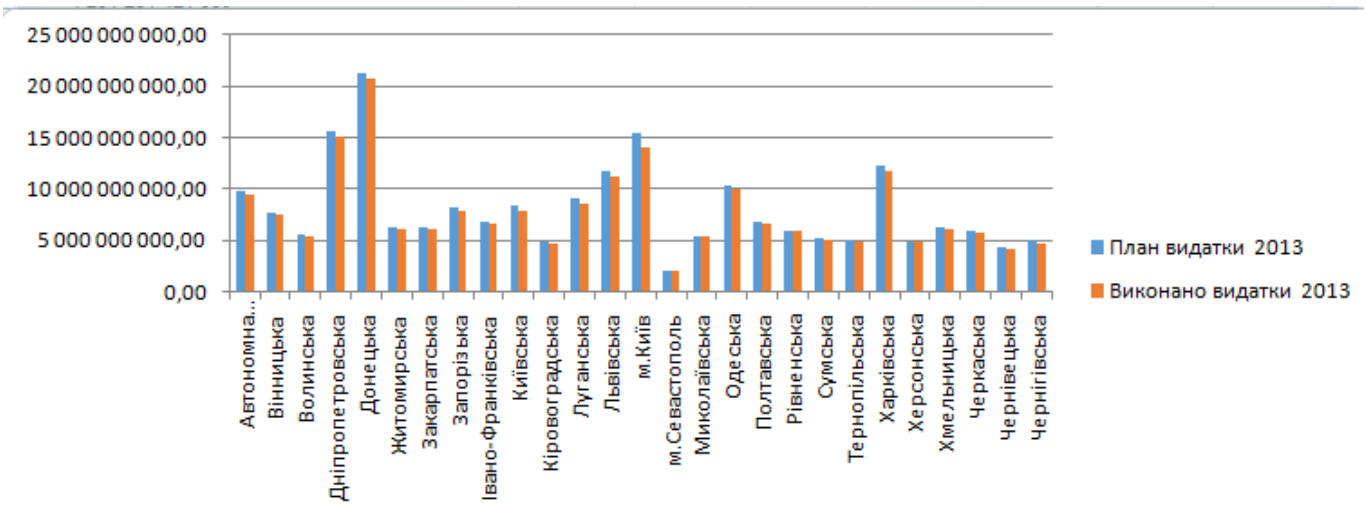


Головними полями для заповнення даних є:

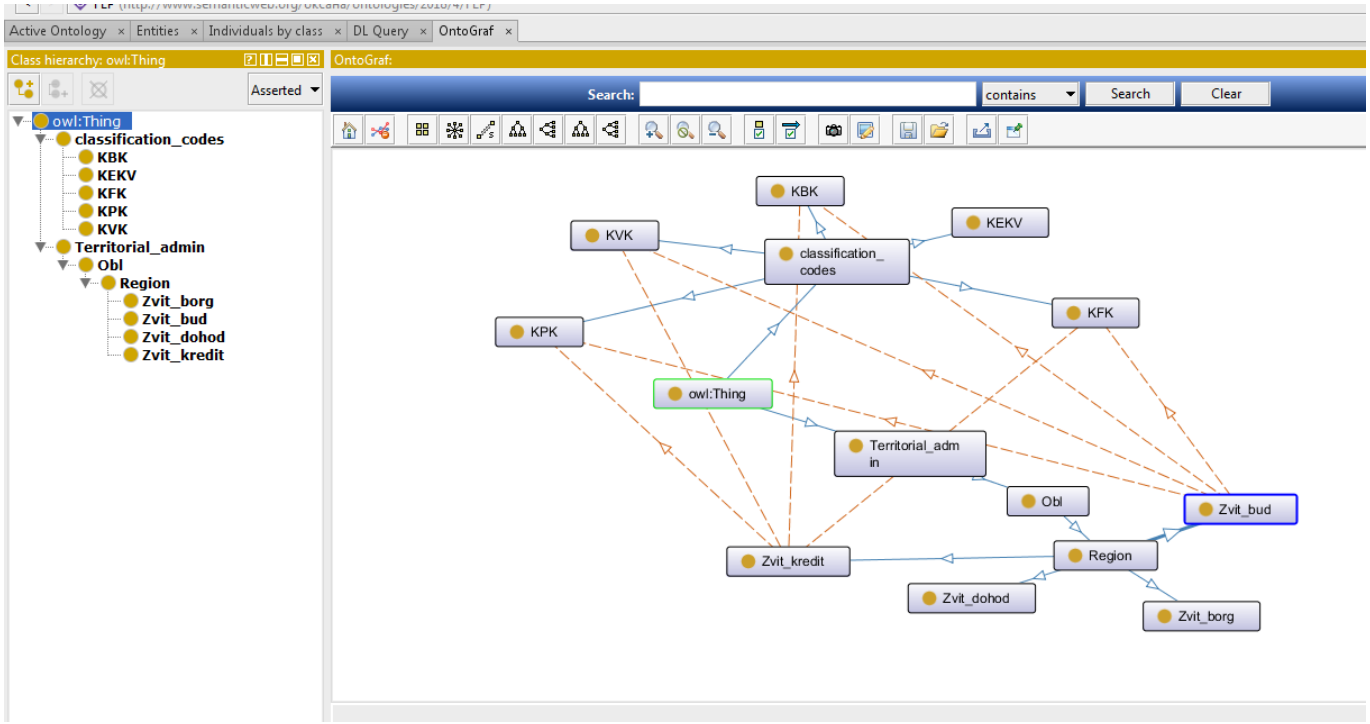
- teritory – поле призначене для введення адміністративної одиниці для якої буде проходити аналіз
- begin year, month – початковий рік, місяць аналізу
- finish year, month – кінцевий рік місяць аналізу
- bugdet executed on – відсоток виконання бюджету
- planned cost – заплановані витрати
- planned revenue –заплановані доходи



Система має можливість аналізувати як групу адміністративних одиниць, так і кожну окремо. Вибрані дані можуть бути конвертовані в Excel з автоматичною побудовою відповідної діаграми.



У онтологічному підході дані матимуть наступний вигляд



5.2 Висновки до розділу

В даному розділі було проведена презентація роботи з програмоюта її основний функціонал. Програма має надає такі можливості:

- завантаження файлу Excel з комп'ютера користувача;
- створення онтології із даними користувача;
- аналіз виконання показників бюджету;
- можливість перегляду даних онтології у структурованому форматі;
- можливість відкриття тієї ж самої онтології за допомогою Neo4j;
- можливість побудови діаграм у Excel;
- підрахунок відсотку виконання бюджету;
- можливість аналізувати як групу адміністративних одиниць, так і кожну окремо

ВИСНОВКИ

Було виконано дослідження методів і програмних продуктів для роботи з онтологіями, проаналізовано їх недоліки та підібрано найзручнішу платформу - neo4j для реалізації поставленої задачі.

Результатом виконання дипломної роботи стала програма, яка забезпечує аналіз виконання показників бюджету.

Розроблений додаток надає можливість, імпортувати і експортувати дані у форматі файла Excel, робити аналіз виконання бюджету по адміністративних одиницях за певний період часу, переглядати і доповнювати онтологію у середовищі програмного продукту neo4j.

Необхідними можливостями, які має забезпечувати модуль є :

- завантаження файлу Excel з комп’ютера користувача;
- створення онтології із даними користувача;
- аналіз виконання показників бюджету;
- можливість перегляду даних онтології у структурованому форматі;
- можливість відкриття тієї ж самої онтології за допомогою Neo4j;
- можливість побудови діаграм у Excel;
- підрахунок відсотку виконання бюджету;
- можливість аналізувати як групу адміністративних одиниць, так і кожну окремо

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is an Ontology? [Електронний ресурс] – Режим доступу до ресурсу: http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.

2. OWL 2 Web Ontology Language RDF-Based Semantics (Second Edition) [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3.org/TR/owl2-rdf-based-semantics/.

3. Triples in RDF/XML [Електронний ресурс] – Режим доступу до ресурсу: https://www.iro.umontreal.ca/~lapalme/ForestInsteadOfTheTrees/HTML/ch07s01.html.

4. Тауберер Д. Краткое введение в RDF / Джошуа Тауберер. – 2005.

5. Муромцев Д. И. Онтологический инжиниринг знаний в системе Protégé / Д. И. Муромцев. – Санкт - Петербург, 2007. – 62 с.

6. Рихтер Д. CLR via C# Программирование на платформе Microsoft .NET Framework 2.0 на языке C# / Джеффри Рихтер. – Киев: Питер, 2008. – 656 с. – (2).

7. Троелсен Э. Язык программирования C# и платформа .NET 4.5 / Эндрю Троелсен. – Киев: вильямс, 2014. – 1312 с. – (6).

8. «Плоский дизайн»: с чего начать? Пять основных принципов Flat дизайна [Електронний ресурс] // powerbranding – Режим доступу до ресурсу: http://powerbranding.ru/design/flat-design-june13/.

9. dotNetRDF An Open Source .NET Library for RDF [Електронний ресурс] – Режим доступу до ресурсу: http://www.dotnetrdf.org/.

10. Сміт Б. Drop-Down Controls Revisited [Електронний ресурс] / Бред Сміт. – 2012. – Режим доступу до ресурсу: https://www.brad-smith.info/blog/archives/477.

11. Фаулер М. GUI Architectures / Мартін Фаулер. // All Sections ThoughtWorks. – 2006.

12. Бодягін І. Model-View-Controller в .Net / Іван Бодягін. // RSDN Magazine. – 2016. – №2.

13. Приёмы объектно-ориентированного проектирования. Паттерны проектирования / Э.Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес. – Харьков: Питер, 2001. – 368 с.

14. Скіт Д. C# in Depth / Джон Скіт., 2013. – 582 с. – (3).

15. Медведєва В. М. Транслятори: лексичний та синтаксичний аналізатори / В. М. Медведєва, В. А. Третяк. – Київ: НТУУ «КПІ», 2012. – 159 с.

Додаток 1

Аналіз виконання показників бюджету на основі онтологічного підходу з використанням платформи Neo4j

Специфікація

УКР.НТУУ“КПІ”.ТР5158_19Б

Аркушів 2

2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ«КПІ ім. Ігоря Сікорського».ТР5158_19Б 81-1	Записка	Пояснювальна записка
Компоненти		
УКР.НТУУ«КПІ».ТР5158_19Б 12-1	Текст програмного модулю	
УКР.НТУУ«КПІ».ТР5158_19Б 13-1	Опис програми	

Додаток 2

Аналіз виконання показників бюджету на основі онтологічного підходу з використанням платформи Neo4j

Текст програмного модулю

УКР.НТУУ“КПІ”.ТР5158_19Б 12-1

Аркушів 6

2019

```
package com.university.controller;


import com.university.Model.TerritorialUnitSearchForm;
import com.university.domain.Revenue;
import com.university.domain.TerritorialUnit;
import com.university.repositories.RevenueRepository;
import com.university.services.CostService;
import com.university.services.RevenueService;
import com.university.services.TerritorialUnitService;
import com.university.services.impl.ExcelReport;
import jxl.write.WriteException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.io.IOException;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.List;

@Controller
    @RequestMapping("/territorialUnit")
public class TerritorialUnitController {

    private final TerritorialUnitService territorialUnitService;
    private final CostService costService;
    private final RevenueService revenueService;
```

```
    @Autowired
    public TerritorialUnitController(TerritorialUnitService territorialUnitService,
                                     CostService costService, RevenueService revenueService) {
        this.territorialUnitService = territorialUnitService;
        this.costService = costService;
        this.revenueService = revenueService;
    }


        @GetMapping(value = "/findTerritorialUnitByName")
    public String ewrwr(Model model) {
        model.addAttribute("searchFormOrderStatistic", new TerritorialUnitSearchForm());
        return "index";
    }

    @GetMapping(value = "/getAllTerritorialUnit")
    public String allTerritorialUnit(Model model) {
        model.addAttribute("territorialUnits", territorialUnitService.allTerritorialUnits());
        return "index";
    }

    @GetMapping(value = "/getAllTerritorialUnit1")
    public String allTerritorialUnit1(Model model) {
        model.addAttribute("territorialUnits", territorialUnitService.allTerritorialUnits());
```

```
            return "index";
        }

        @PostMapping(value = "/findTerritorialUnitByName")
        public String findBySearchTerm(@RequestParam String name,
                                       @RequestParam(name = "start_date")
@DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate start_date,
                                       @RequestParam(name = "end_date")
@DateTimeFormat(iso = DateTimeFormat.ISO.DATE) LocalDate end_date,
                                       Model model) {
            model.addAttribute("cost", costService.getValueAllCostByUnit(name,
start_date, end_date));
            model.addAttribute("revenue",
revenueService.getValueAllRevenueByUnit(name, start_date, end_date));
            model.addAttribute("budgetAnalyze",
territorialUnitService.getAnalyzeBudget(name, start_date, end_date));
            System.out.println(territorialUnitService.getAnalyzeBudget(name,
start_date, end_date));
            return "index";
        }


        @PostMapping(value = "/findTerritorialUnitByName")
        public String findBySearchTerm1(@ModelAttribute("searchFormOrderStatistic")
TerritorialUnitSearchForm territorialUnitSearchForm,
                                        Model model) throws IOException,
WriteException {

            ExcelReport excelReport = new ExcelReport();
            excelReport.createReport();
            System.out.println("all good");
            System.out.println(territorialUnitSearchForm.getName());
```

```
                System.out.println(territorialUnitSearchForm.getStart_date());
                System.out.println(territorialUnitSearchForm.getEnd_date());
                model.addAttribute("cost",
costService.getValueAllCostByUnit(territorialUnitSearchForm.getName(),
territorialUnitSearchForm.getStart_date(), territorialUnitSearchForm.getEnd_date()));
                model.addAttribute("revenue",
revenueService.getValueAllRevenueByUnit(territorialUnitSearchForm.getName(),
territorialUnitSearchForm.getStart_date(), territorialUnitSearchForm.getEnd_date()));
                model.addAttribute("budgetAnalyze",
territorialUnitService.getAnalyzeBudget(territorialUnitSearchForm.getName(),
territorialUnitSearchForm.getStart_date(), territorialUnitSearchForm.getEnd_date()));
                model.addAttribute("territorialUnits",
territorialUnitService.allTerritorialUnits());
                System.out.println( territorialUnitService.allTerritorialUnits());

                return "index";
            }
        }

        package com.university.repositories;

        import com.university.domain.Cost;
        import com.university.domain.TerritorialUnit;
        import org.springframework.data.neo4j.annotation.Query;
        import org.springframework.data.neo4j.repository.Neo4jRepository;
        import org.springframework.data.repository.query.Param;
        import org.springframework.stereotype.Repository;
        import org.springframework.web.bind.annotation.RequestMapping;

        import java.util.Collection;

        @Repository
```

```
public interface TerritorialUnitRepository extends Neo4jRepository<TerritorialUnit, Long> {

    @Query("MATCH (n) RETURN n")
    Collection<TerritorialUnit> getAll();

    @Query("MATCH (m:TerritorialUnit {name: {territorialUnit} } )-[:PARENT*0..15]->(r)<-[:RECEIVED]-(e:Revenue) RETURN m,r,e")
    Collection<TerritorialUnit> getValueAllRevenueByUnit(@Param("territorialUnit") String territorialUnit);

    @Query("MATCH (t:TerritorialUnit {name: {territorialUnit} } )-[:PARENT*0..15]->(p)-[:PAY]-(c:Cost) RETURN t,p,c")
    Collection<TerritorialUnit> getAllCostUnit(@Param("territorialUnit") String territorialUnit);


}
```

Додаток 3

Аналіз виконання показників бюджету на основі онтологічного підходу з використанням платформи Neo4j

Опис програмного модуля

УКР.НТУУ“КПІ”.ТР5158_19Б 13-1

Аркушів 4

2019

АНОТАЦІЯ

Метою дипломної роботи є розробка програмного додатку, який дозволить користувачу проводити аналіз виконання показників бюджету на основі онтологій

У ході роботи проаналізовані засоби створення онтологій – Protégé, GraphDB , Neo4j та зроблено порівняльну характеристику. В роботі обґрунтовано використання онтологічного підходу для зберігання даних показників виконання бюджету. В якості подальшого розвитку системи планується створення графічного інтерфейсу для заповнення даних та встановленню між ними зв’язків

ЗМІСТ

1 ВІДОМОСТІ ПРО ПРОГРАМНИЙ МОДУЛЬ

Даний програмний модуль розроблено у середовищі IntelliJ IDEA, , використовуючи типізовану мову програмування Java, не типізовану мову програмування JavaScript, бібліотеку Thymeleaf, платформу Neo4j та деякі додаткові бібліотеки.
Програма призначена для аналізу виконання показників бюджету

1.1. Опис логічної структури

Було розроблено багатоплатформний програмний продукт, основною задачею якого є побудова аналізу виконання показників бюджету.

Нажаль, реляційні бази даних, які традиційно використовують для задач такого типу, не мають графічної візуалізації зв'язків між класами предметної області. Аналіз виконання показників бюджету зручно проводити маючи графічну візуалізацію зв'язків та залежностей між окремими даними. Використання онтології якраз допоможе вирішити цю проблему. Онтологія надає такі переваги для реалізації поставленої задачі:

- візуалізація графу даних;
- аналіз за допомогою предикатів;
- валідація вхідних параметрів;
- представлення ієрархічної структури на різних рівнях вкладення;
- зручне підтримання актуалізації даних.

1.2. Вхідні та вихідні дані

Вхідними даними для системи є показники бюджету введені користувачем або імпортовані із файлу Excel

Вихідними даними є проведений аналіз виконання показників бюджету

2 ВИКОРИСТАНІ ТЕХНІЧНІ ЗАСОБИ

Програмний модуль було протестовано в браузері Google chrome 10.0 на персональному комп’ютері, який працює на базі процесору 2,6 GHz Intel Core i5 та має 16 Гб оперативної пам’яті. Розроблене програмне забезпечення є кросбраузереним та кросплатформенним, що дозволяє запускати його на комп’ютерах будь-якої потужності та в будь-яких сучасних браузерах.

Додаток 4

Аналіз виконання показників бюджету на основі онтологічного підходу з використанням платформи Neo4j

Апробація

УКР.НТУУ“КПІ”.ТР5158_18Б 13-1

Аркушів 4

2019


www.konferenciaonline.org.ua
Міжнародна наукова
інтернет-конференція
Інформаційне суспільство:
технологічні, економічні
та технічні аспекти становлення
(випуск 38)
Частина 1
ISSN 2522-932X
7 травня 2019 р.
Тернопіль
2019

Частина 1

Секція 1. Інформаційні системи і технології

ВИКОРИСТАННЯ ОНТОЛОГІЧНОГО ПІДХОДУ ДЛЯ АНАЛІЗУ ВИКОНАННЯ ПОКАЗНИКІВ БЮДЖЕТУ

У сучасному світі є необмежений доступ до великої кількості інформації. Швидкий розвиток інформаційних технологій приводить до появи інформаційних систем великого масштабу. Багато аналітичних систем

використовують велику кількість різнорідної інформації.

Вже давно існує проблема організації збору потрібної інформації для її подальшого аналізу. До таких задач можна віднести задачу моніторингу виконання показників бюджетів. Фактично, це процес збору та аналізу даних про доходи та видатки, згідно закону України «Про державний бюджет» .

Звіти про виконання державного бюджету на регіональному рівні надходять до Держказначейства України. Автоматизоване введення даних про виконання бюджетів всіх рівнів передбачає використання відповідних форм звітів Державної казначейської служби України, представлених у форматі MS Excel. Звітність про виконання бюджетів всіх рівнів є оперативною, місячною, квартальною та річною. Всі ці звіти проходять етапи збору інформації від окремих підприємств до районної звітності в області, відомства, міністерства. Існують також звітності про виконання окремих державних програм та ін.

Розподіл показників є досить розгалудженим, як за територіальними так і за фінансовими показниками, та враховує обсяги фондів та їх цільове призначення зі своїми особливостями та правилами.

Для зберігання та обробки великих об’ємів структурованої інформації традиційно використовують реляційні сховища даних. Але залишається проблема об'єднання даних, отриманих з різних джерел з метою їх об'єднаного подання користувачеві

Вирішенням проблеми формалізації даних може бути онтологічний підхід. Такий підхід досить часто використовується для об’єднання знань предметних областей в єдину інформаційну структуру.

Онтологія - це один із методів візуалізації та побудови зручних графів для дослідження даних. Ця технологія забезпечує хорошу взаємодію різних специфікацій метаданих за допомогою спільної семантики, структури та синтаксису. Мова веб-онтологій або OWL додає більш потужні інструменти моделювання до RDF і RDFS. OWL забезпечує перевірку на узгодженість та на відповідність певним умовам. Наприклад, чи є якісь логічні невідповідності або чи існують класи, які не можуть мати екземплярів. Також OWL забезпечує класифікацію об’єктних типів.

На рисунку наведено частину онтології системи моніторингу фінансово економічних показників бюджету виконаної у середовищі Protégé.

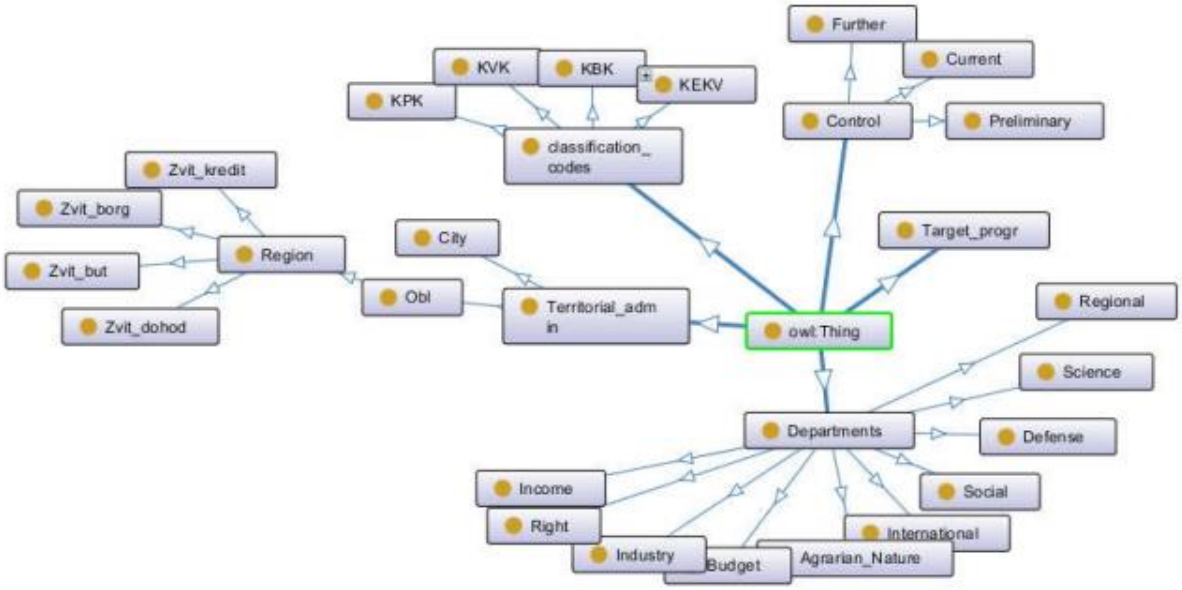


Рисунок 1. Онтологія моніторингу фінансових показників виконання бюджетів

Аналіз і візуалізацію показників бюджету складно провести за допомогою реляційних баз даних, тому онтологія, яка може надати такі переваги для реалізації поставленої задачі:

- візуалізація графу даних;
- аналіз за допомогою предикатів;
- валідація вхідних параметрів;
- представлення ієрархічної структури на різних рівнях вкладення;
- зручне підтримання актуалізації даних.

Самі ж звітні дані знаходяться в таблицях, які пов’язані з класами онтології. За допомогою фреймворка «Ontop» можна під’єднати онтологію до таблиць бази даних. В результаті можливе виконання запитів до онтології, які автоматично будуть отримувати дані з таблиць. Для того, щоб фреймворк «Ontop» правильно генерував і виконував запити до бази даних потрібно як можна точніше описати всі властивості об’єктів і типів даних з онтології і створити всі потрібні «маппінги» для всіх властивостей.

Таким чином можна проводити автоматичний збір та обробку інформації на основі звітів про хід виконання держбюджету, що сприяє контролю за ефективністю використання коштів.